BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

z e t e r a

Zetera White Paper on

µSAN™

A technical dissertation on a simple IP architecture and application level protocol to share block-oriented resources (including hard drive storage) across an IP (inter-network protocol) network.

Version 0.35

Author:
Thomas E. Ludwig
Thomas D Hanan
VP Engineering, Zetera Corporation

December 2002

Revision History

Version 0.1	October 2002	Original Release
Version 0.2	December 2002	Added clarification through the document Changed the LBA field size from 4 to 5 Bytes to cover Disk Spanning to 512 Terabytes Added Multicast for Spanning and RAID Support Added "How DHCP is used in μSAN™"
Version 0.3	December 2002	More clarification Added UPnP Support Added clarification on using multicast with the GO class of commands. Modified the ACK and ERROR command to include source IP (from the destination μSAN™) for exception handling in GO multicast operations. Added Authentication and Security Protocol
Version 0.35	December 2002	Updated and enhanced the Authentication and Security Protocol (more detail)

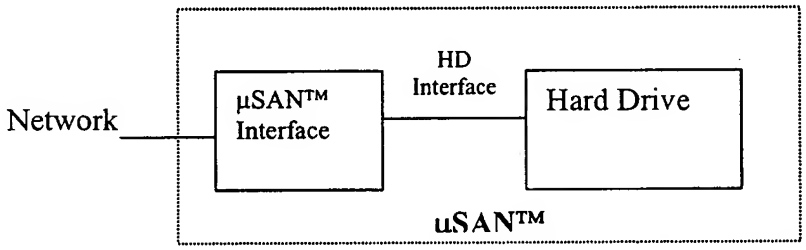
Contents

Overview

This White Paper identifies the problem and solution dynamic for simple, low-cost block I/O IP based storage hereafter referred to as µSAN™. Using common IP (Internetworking Protocol) technology, the µSAN™ may be attached to any standard underlying network without change to the established infrastructure.

The µSAN™ is an IP connected storage device that is not burdened by the file system nor tied to a specific OS. Issues of security may be handled through existing IP protocol and authentication is performed at the application layer. The file system is defined by the "Requester" and can be as complex or simple as necessary.

This document defines a "**Requester**" as an appliance device or computer that uses the services of a µSAN. In this nomenclature, the µSAN™ is a "**Responder**". These semantics were adopted to avoid confusion with the client/server architecture so pervasive in the IP environment. The Requester/Responder architecture that the µSAN™ belongs to is more universal allowing it to conform to either a peer-to-peer or a client/server network.



The µSAN™ interface performs the following functions:

- Interfaces with the network's physical layer. This may be Ethernet, 802.11x or any physical connection that interfaces to the IP layer.
- Provides an Application Layer Protocol. The µSAN™ protocol sits between the IP transport layers (TCP & UDP) and the HD interface protocol (ATA). It should be understood, however, that the µSAN™ protocol DOES NOT tunnel ATA protocol.
- Maintains and interfaces the partitions to the Requester providing for a multiplicity of logical units with one drive.
- Executes logical read and write block functions. I/O to the drive is based upon logical blocks.
- Utilizes a self contained "Naming" convention for storage resource discovery, instantiation and IP maintenance.
- Utilizes DHCP for IP address allocation. Alternatively "Auto IP" may be used as indicated in the UPnP specification.
- Provides for data integrity through IP and storage verification standards.

Perceived Problem

The μSAN™ is a general solution platform created to address the following perceived problems:

- The emergence of new low-cost sensitive consumer electronics platforms can benefit from the mass storage offered by hard-drive storage and yet cannot afford the component cost of a dedicated hard-drive.

- There is a need for simple expandable storage in the home/SOHO environment that is not constrained by a complicated client/server protocol.

- Network storage is expensive either through costly NAS servers or costly SAN servers. What we need today is low-cost, efficient storage accessible through IP.

The Need for Low-Cost Consumer Electronics Storage

Consumer electronics is increasingly forced to address the need for mass storage due to the transport of large amounts of video and audio data. We see this in the desire for PVR (personal video recorder) and MP3 jukebox functionality. However, at the same time, the consumer/manufacture/supplier cannot overcome the cost obstacles to achieve high unit volume. The lowest cost of a hard drive is fixed, and yet the cost per gigabyte decreases significantly every year as aerial bit density increases. A consumer electronics device, however, does not need a $100, 100 gigabyte solution so much as it needs a $10, 10-gigabyte solution or a $1, 1-gigabyte solution. An obvious solution is to disaggregate the drive creating separate SKUs and to amortize the cost of the storage across many low-cost device applications. This is precisely what a μSAN™ does. A single μSAN™ can service the needs of many consumer electronics devices. The μSAN™ protocol allows the requesting device to be as simple or complex as the application needs further lowering the cost. The μSAN™ itself is a low-cost product. The devices that address the video/audio world through the μSAN™ are no-longer burdened with the problems of storage cost, upgrade, service and scalability.

Shared Storage in the Home/SOHO Environment

Shared storage is currently constrained by the Client/Server protocol inherent in the infrastructure. The sharing of the resource is accomplished through the computer's OS that "owns" the drive. This places a dependency upon that computer's behavior as to the reliability of the connection. In other words, to share that resource a second computer (or device) must rely upon the integrity of the primary computer including its user profile, power-on, and OS status. A μSAN™ may be connected to the network and by connecting at the block level

behave in an equitable peer-to peer relationship with all of the computers connected to it. It does NOT rely upon any dominant connection. Shared storage is "fairly shared" and therefore solely answerable to the computer that requests it.

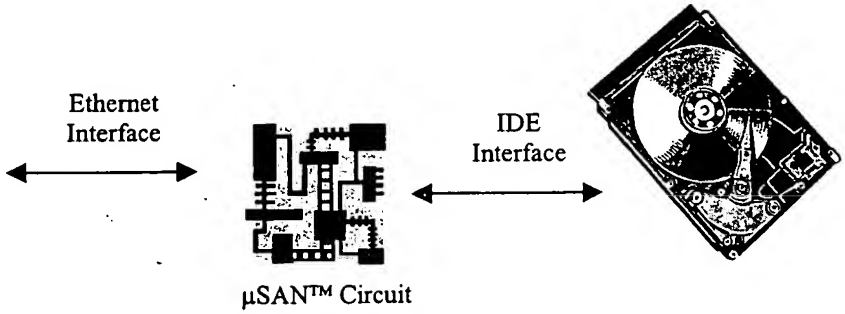
The need for Low-Cost Network Storage

Shared storage is today ONLY accessible through an OS. The OS maintains the storage resources, authenticates and secures the access and maintains elements within the stack. For a NAS it also maintains a file system and application protocol for file transfer. The storage element talks to the network THROUGH the OS. All of this mandates a costly implementation for network storage, not to support the storage, but to support the OS. The μSAN™ protocol complements the IP to provide the same functions as the OS removing the need for such an OS and therefore has the lowest possible cost of network storage implementation.

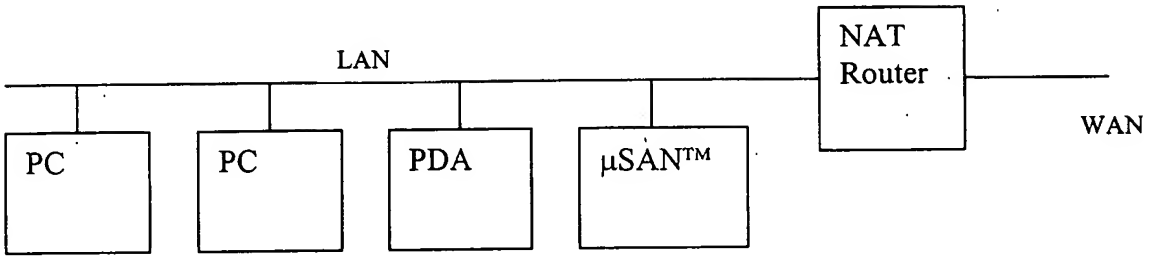
Future Adaptation

It is perceived that a simple adaptation of IP connected block level storage could migrate itself from add-on storage at the network level to using the IP backbone to replace direct attach as well as enterprise system fabrics. The IP protocol with an Ethernet network is the most pervasive computer communications interconnect in use. As Ethernet becomes faster, wireless and more QoS (Quality of Service) sensitive, the IP protocol provides for an inter-connection virtually without limitation. The IP stack is a well-known and familiar discipline that can provide for the future and it is well established.

Today's implementation of the µSAN™ would be as an add-on PCB or module to the existing ATA Drive with an IDE interface such as:



For example, this drive would interface to the computer network as follows:



By integrating the circuitry onto the drive, it is perceivable that the same implementation may expand to the desktop and enterprise markets as well becoming a direct-connect onto the desktop and part of a fabric in the enterprise. This evolution would allow the hard drive component to be both integrated to the PC as well as easily upgraded through the network with the software treating it in either case as an IP target component.

Competitive Alternatives

At a gross level, one should consider the NAS (Network Attached Storage) as an architectural alternative. This robust (enterprise level) solution plays well in an IT environment, where security and file management are paramount. However, the burden of this architecture is too much to bear in the consumer marketplace. The burden is inherent in the definition of the NAS itself. It is a file-oriented architecture. This mandates a compatible OS. This mandates a server that is expensive and requires IT level support.

Block level shared storage exists in today's SANs in three protocols - iSCSI, iFCP and FCIP. Each of these protocols has been designed for the enterprise market to exist in a fabric. They were not designed for the consumer market and cannot accommodate sharing the storage in a mutually exclusive manner to the requesting device. In addition, the protocols are verbose and hence expensive to implement at the device level.

Shared storage on a PC has the inherent disadvantage of dependence upon the PC being available (powered-on, shared, OS running). In addition, the PC does not scale its storage easily (adding drive resources). This solution also demands that the file and protocol structures be compatible with the PC including NetBios over IP, FAT32 and Master Browser (in the Windows world).

Perceived Intellectual Property

- Allocation of logical drive partitions to an IP address.
- Name resolution
- Block validation
- Simple protocol that works through NATs
- Semaphore protocol
- A common protocol for both TCP and UDP implementations
- Use of Multicasting to provide Spanning and RAID like functionality
- Atomic protocol simplifying authentication and resolving connectionless network protocol issues.

Technical Issues to be Resolved

- Use IP to share the drive without the encumbrance of an OS
- Simple protocol that can be implemented by the Requester at a low-cost to the appliance.
- Simple protocol that works well both in a routed and non-routed network environment.
- Protocol that can perform transfers across a NAT (Network Address Translation) protocol router.
- Partition allocation protocol including a Name Resolution that is peer-to-peer in nature.
- Partition Release
- Discovery of a μSAN™ on the network
- Reliability
- Authentication and Security
- QoS
- Multi-session Access including lock functionality
- Sustaining
- Low Cost Architecture
- Little Endian vs. Big Endian
- Use of Multi-Cast IP for RAID and Spanning
- Work in a UPnP Network Environment

IP to Partition Resolution Protocol

It is desired that each Requester be allocated a partition that is mutually exclusive to other partitions on the drive. A Requester may "own" the partition, "visit" the partition or "rent" the partition. The μSAN™ performs this by allocating a logical partition to a unique IP address and Name. The Requester may then share this allocation with other Requesters on the network (passing the Name and Token) at its discretion. This process is limited only by how many resources each μSAN™ may handle in its code.

The μSAN™ is IP plug and play like on the network and as such conforms to the DHCP (Dynamic Host Configuration Protocol) standard for IP address allocation. However, because it is always a "Responder" (never an initiator itself) there is no need for it to use to the DNS (Domain Name Server) address provided by the DHCP server. The μSAN™ does not use the DNS protocol directly. (See "Use In A Server Environment" for DNS use.) Alternatively, the μSAN™ may use the Auto IP algorithm under UPnP in those networks where a DHCP server is not available. (See "UPnP Consideration".)

Each μSAN™ has a singular Root IP address that is used to communicate with the drive and perform such functions as Partition Allocation. This Root IP is allocated at the power-on boot through a DHCP allocation and is discovered by the Requester through a μSAN™ FIND packet. (At this time it is sufficient that the Requester is able to obtain the Root IP to interrogate the drive).

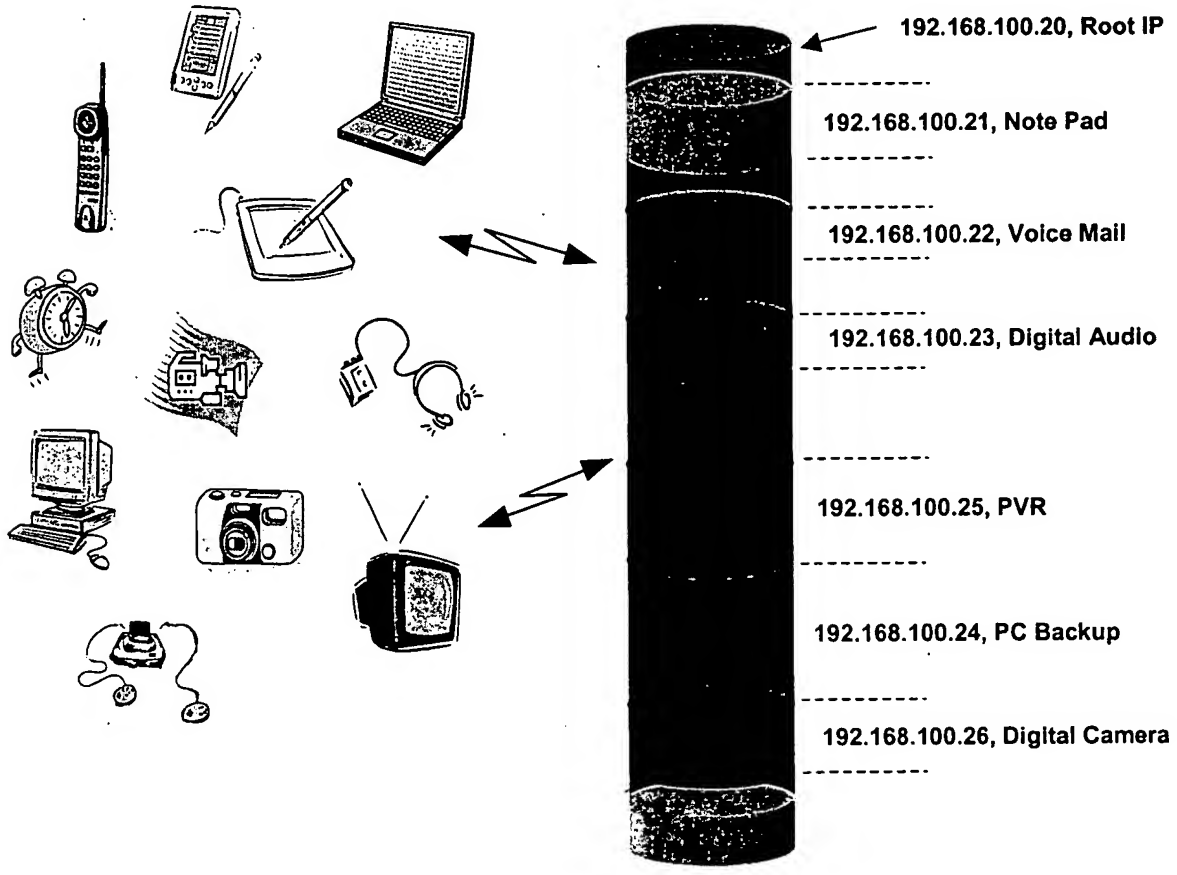
Rules:

- The μSAN™ has one MAC address, associated with multiple IP addresses.
- Each partition has a unicast IP Address allocated through DHCP or Auto IP.
- Each unicast IP Address has a unique Name (except for the root IP).
- Requesters address the drive through the Name.
- Each partition has a set of logical blocks that nominally start at 0. In the case of Spanning, this number will be an offset value.
- There is no "quota". If you run out of drive space, buy another drive.
- The creation of a partition is made with the "owner" supplying the appropriate Name and a Token. The μSAN™ will get an IP address from the DHCP server for the partition. The Requester will also supply a user-friendly character string that identifies the partition to the Requester. This character string is NOT a function of partition authentication.
- When a partition is released, the IP/Name is released. The partition is "scrubbed" and the physical blocks become available for future partition allocations.
- Partitions may span physical blocks.
- There are no rules about how a μSAN™ deals with partition block fragmentation or bad blocks.

Partition Allocation

1. Requester discovers μSAN™ root IP. (See μSAN™ Instantiation on Network)
2. Requester discovers unused drive space and personality characteristics.
3. Requester requests an allocation by providing a Name, Token, size and authentication parameters (see Authentication and Security protocol) plus any available personality variables. It also provides a Partition Name character string.
4. The μSAN™ goes through DHCP to request an IP address for the new partition and assigns it to the NAME. It also passes the NAME back to the DHCP server as a "unique ID" for future association. In a network absent of a DHCP server, Auto IP as described under UPnP may be used.
5. Requester requests access to the partition through the μSAN™ Name Service, which resolves to the newly assigned IP address.

The result of this allocation is shown below where multiple Requesters have access to multiple partitions on the same drive through individual IP addresses:



Partition Release

1. Requester issues a special Partition release command with the appropriate authority.
2. The μSAN™ rewrites, (scrubs), the logical block removing any residual data and requests a release of the IP through the DHCP server, or drops the Auto IP address. Any associated Multicast IP address is also no longer supported and the μSAN™ will not respond to an IGMP Query to this Multicast IP.
3. The μSAN™ places the removed and cleaned logical blocks back into the allocation pool for future reallocation.

How DHCP is Used in the μSAN™

There are two hierarchal levels for DHCP allocation within a single μSAN™ component. The first is at the "root IP" while the rest are associated with each partition. It is important to note the distinction. Root IP is allocated at power-on without client involvement whereas a client must first allocate a partition before partition IP allocation can take place. Most IP compliant devices have a one-to-one relationship between IP and hardware addresses and the DHCP server typically associates this in a table. The μSAN™, however, has one hardware address associated to many IP addresses and must used the optional "unique client identifier option" in the DHCP protocol under section 9.14 of RFC 2132. (Note, If no DHCP server is present on the network then Auto IP allocation will be performed as specified under the UPnP structures.)

Consider the 'Root IP"

After "power-on", a μSAN™ will use DHCP to allocate an IP to the root for initial communication in Discovery. The unique identifier used for this allocation, as defined in RFC 2131,2132 maybe either the hardware address or based upon the components serial number.

Consider the partition IP

Initial allocation for IP at the partition level occurs after a client requests a partition allocation from the μSAN™. The client will pass to the μSAN™ a unique Name used in the Name Resolution protocol (see μSAN™ Name Resolution Protocol). This Name is also used in the optional "Client ID" field of the DHCPREQUEST command in DHCP. This will provide not only a unique identifier for each partition IP address, but will also provide a lookup table for IP association to Name at the DHCP server. This solves the problem with one hardware address and at the same time provides a maintenance table.

Partition Personality Variables

Each partition may have personality variables that are established at the time of Partition Allocation. These variables are dependent upon the μSAN™ model itself. The following is a list of potential variables:

- Write Once – If set, the data, once written, may never be erased. This option could be used in backup/archival situations where the validity of the data is paramount. Security video, legal documents, time stamped accounting files, etc.
- QoS – A variable may be set here that guarantees the bandwidth to/from this partition. This variable would work in conjunction with other QoS parameters in the network. At the IP level, this would include the ToS (Type of Service) bits in the IP header.
- Extended block size. Typical block size is 512 bytes. Extended block size is 530 bytes. Used to accommodate embedded LBA and/or CRC at the Requesters discretion.

Authentication and Security Protocol

As a transport protocol, security within the µSAN™ protocol is focused on ensuring the integrity of the transport. This focus allows µSAN™ to remain compatible with the diverse assortment of payload protection and DRM (Digital Rights Management) methodologies expected to be in use during the life of the µSAN™ protocol.

The µSAN™ authentication protocol itself does not transfer data, and is thereby exempt from federal & international limits on the strength of the algorithms and code lengths used. This crucial distinction allows µSAN™ compatible devices to support simple lightweight algorithms, which provide "STRONG" authentication.

As with the rest of the µSAN™ protocol, the authentication protocol is designed to allow individual devices to implement a range of functionality while remaining compatible with the protocol.

The range of µSAN™ authentication functionality can be broken into four (4) key methodologies providing increasing levels of protection from malicious or accidental packet corruption.

- Requestor IP Address
- Requestor MAC ID (Harder to Spoof, But Possible)
- LBA DAC (Data Authentication Code, Based on Originator Token)
- Payload DAC (Based on originator Token)

Combining the four (4) authentication methodologies provides devices transporting data with the ability to overcome a wide range of real world packet authentication problems that remain un-addressed in other protocols.

The µSAN™ authentication protocol relies upon several tenants of a good security system.

- Use seed data difficult or impossible to snoop. (Time of Manufacture)
- Never exchange "core" seed data in the clear.
- Limit the life of any valid code to four orders of magnitude less than the time required for a statistical or algorithmic attack.
- Limit the usefulness of any single algorithmic code attack to a single transport.
- Allow for the detection and isolation of failed code attack attempts.
- Support Dynamic Code lengths. (Future Proof)

In addition to the above requirements the authentication codes used by the µSAN™ authentication protocol needed to be tolerant of out of order reception and lost packets.

As such the μSAN™ TAP (Transport Authentication Protocol) uses a synchronized window algorithm that allows for a device selectable window of valid TAC's (Transport Authentication Codes). Each successfully authenticated transport packet moves the window forward thereby aligning the window of valid TAC's around the statistical mean of the packets in flight. Packets delayed sufficiently to fall outside the window are treated as lost packets by the devices at both ends. If all packets received within a device configurable window are invalid the receiving device returns a TAP NAK indicating loss of TAP syncronization. The TAC length is such that the number of valid TACs within the window is statistically insignificant as compared to the length and spread of the TAC code.

The synchronization algorithm denies the attacker vital information by not providing NAK for "BAD" code packets until synchronization is lost. In this way the algorithm prevents attackers from denying service to valid devices, which can maintain synchronization in the presence of "BAD" packets.

The device synchronization and authentication code engines are implemented as systolic algorithms to minimize the processor cycles required and future proof the length of the codes. Instantaneous processing requirements can be further reduced by pre-calculating the codes in the background.

At the time of Partition Allocation, the Requester establishes the level of authentication and security that it desires. Because in a peer-to-peer architecture there is no server to validate and issue security, there is a need for some security to be established by the μSAN™. (See Use in a Server Environment for security under a server architecture.)

The Requester establishes for each partition, the Name and a TAT (Transport Authentication Token) . The Name is used for IP address resolution and the TAT is used to establish and maintain synchronization with the TAW (Transport Authentication Window). The Requester may also pass the TAT to another Requester in order to share a common partition. Even when sharing the same partition Devices must maintain a separate TAW for each device they communicate with using TACs.

There are three values used for the four different types of partition operation and may be programmed for authentication as follows.

	Read	*Write*	*Release*
Requester IP	~/AND/OR	~/AND/OR	~/AND/OR
Requester MAC	~/AND/OR	~/AND/OR	~/AND/OR
LBA DAC (TAC)	~/AND/OR	~/AND/OR	~/AND/OR
Payload DAC (TAC)	~/AND/OR	~/AND/OR	~/AND/OR

Each of the variables may be programmed as a Boolean "Don't Care", AND or OR function. For example, to allow a Read with either the IP Address or Mac Address matching as long as the payload Token matches would be programmed as:

Requester IP Addrs = OR
Requester MAC Addrs = OR
Payload DAC = AND

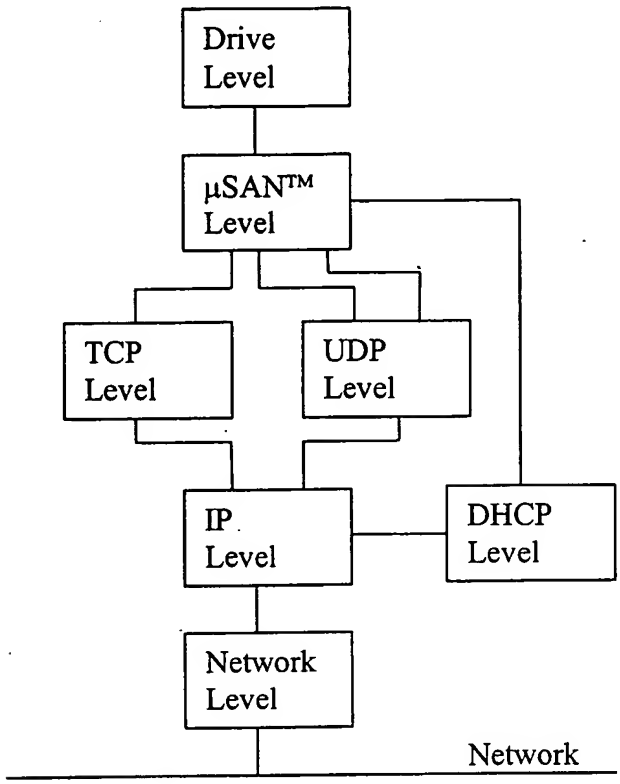
This authentication scheme allows one Requester to write, but many to read. It also allows very tight or loose control over the release process. A partition release is a non-recoverable operation and should have a tighter authentication level. If tokens are used and later forgotten, the partition is lost.

Care must be exercised in the use of the IP and MAC authentication. If a DHCP reallocation occurs, the source IP address may change. If a new source NIC is used, the MAC address will change. The TAC will however always remain valid and may be used minimally to validate the identity of devices that need to change IP or MAC addresses.

Payload security is the responsibility of the Requester. Because the μSAN™ is a block level storage medium, any type of raw data encryption or DRM may be utilized.

IP Stack Impact and Data Reliability at the µSAN™

Consider the IP stack and the impact of the µSAN™ at each level.



- Network Level

  If the network is a LAN such as Ethernet, it must be considered that the µSAN™ will have one MAC (or physical) address and many IP addresses. This does not violate IP protocol and is currently in practice. However, if a PC wishes to emulate a µSAN™, the NIC must also be capable of this operation. There is no impact upon the Requester.

- IP Level

  The only consideration on this level is the need to support the Type of Service bits in the IP header. The high data throughput bit should be set to inform the subsequent routers of the need for high performance. (ARP will need to respond to multiple IP request.) There is no impact upon the Requester.

- DHCP Level

  The DHCP engine in the μSAN™ must be capable of supporting multiple IP address allocations. Normally, this engine handles just one. Each partition in the μSAN™ has a unique IP address that is associated with that partition exclusively. The DHCP engine must return a unique identifier for association to the DHCP server. A common practice is to use the MAC as the unique identifier. But, because we have only one MAC, it is recommended that the Name be used instead. (see "How DHCP is used in the μSAN™")

- TCP/UDP Level

  The μSAN™ will support both UDP as well as the TCP protocol at the transport level. The same application protocol (μSAN™ Protocol) will be connected to either transport protocol. (The μSAN™ application protocol is that simple and generic). This allows a Requester to achieve a very low cost in a non-routed LAN environment by using the UDP transport and yet at the same time be able to use the TCP transport for those needs where the Requester either communicates far outside of the LAN through routers or there is a need for multi-session access. Once again, the Requester makes the choice to meet its needs.

  Note that a common UDP/TCP port needs to be established for μSAN™ initial communication. A separate UDP port needs to be established for μSAN™ Name Services Resolution **(This still needs to be determined.)**

  - UDP

    The UDP (Universal Data Packet) protocol is very simple and generic. The advantages are low overhead and it works well in a stable network environment. The disadvantages include that it is a "connectionless packet delivery" protocol meaning that there is no guarantee of data transfer and packet ordering. There is also no provision for multi-session instantiation. In a Requester/storage need where the sessions are single threaded and the network is stable (such as in a LAN), UDP provides the Requester with the simplest of implementations. The μSAN™ protocol provides for data transfer ACK as well as data ordering.

    UDP is also used for the μSAN™ Name Services resolution and μSAN™ Find protocol. (see Broadcast Protocol)

  - TCP

    The TCP (Transmission Control Protocol) protocol is a "reliable stream delivery" transport protocol that also supports multiple sessions. This protocol establishes a port connection between the Requester and the

μSAN™ that identifies the data transport for that session allowing for multi-threaded operations. The protocol also provides for data delivery and correct ordering by providing the appropriate "handshaking". The TCP protocol would be best used by Requesters that either have multi-threaded storage needs (such as PC storage) and/or communicate through an unstable network environment (such as the Internet). If communication is necessary outside of the LAN, a proxy server must translate μSAN™ Naming Services to an internetwork naming services protocol such as DNS.

- μSAN™ Protocol Level

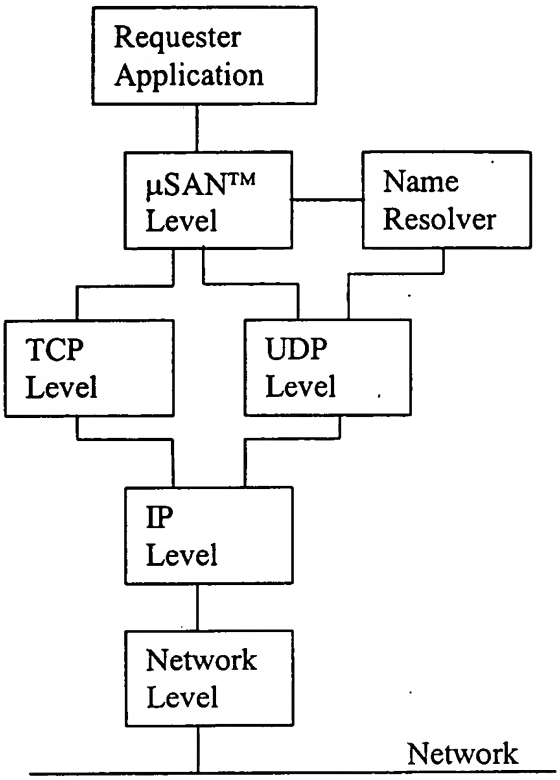
  The protocol at the μSAN™ level performs the following functions:

  - Partition Allocation and Release
  - Reports μSAN™ Status
  - Performs Authentication Verification
  - Translates Logical to Physical Blocks based upon IP
  - Translates μSAN™ Block Transfer protocol into ATA protocol
  - Reports Errors
  - Perform μSAN™ Name Services (see Naming Protocol)
  - Perform μSAN™ Find protocol

  The μSAN™ protocol code must be re-entrant in order to support both multi-sessions within the same partition as well as multiple sessions to different partition/IP pairs. There is a conceivable limit to how many sessions can be open at the same time based upon the implementation resources of the μSAN™. If a μSAN™ cannot open another session, a request is error ACKed until one becomes available. In an environment of many Requesters, a token may be passed between the Requesters outside of this protocol to provide a pair sharing arbitration scheme.

IP Stack Impact and Data Reliability at the Requester

Consider the IP stack and the impact of the μSAN™ protocol at each level.



- Network Level

  Standard.

- IP Level

  Standard.

- TCP/UDP Level

  UDP is used for the μSAN™ Name Services resolution protocol, which by definition is not propagated through a router. If the communication crosses a router, then the μSAN™ Naming Services must be translated by a proxy to an internetwork naming service such as DNS.

  The demands of the Requester application will dictate whether the Requester chooses to use a TCP (reliable stream transport) or UDP (connectionless transport) protocol for the block I/O transfers. The UDP protocol is cheaper and

simpler to implement and would be used in those environments where cost is an issue and the environment is limited to the local area network and single threaded operations.

If multi-session operations are necessary (such as a PC volume) and/or the transport is across an internetwork (many routers), then the Requester would use the more robust TCP transport protocol for block I/O transfer. To the μSAN™, the choice is immaterial.

- μSAN™ Protocol Level

  The protocol at the μSAN™ level performs the following functions:

  - Request and Release Partition
  - Translates requests from the Requester application to block I/O transfers to the μSAN™
  - Performs error exception handling
  - Optionally establishes groups of multicast drives for Spanning and RAID operations

  The μSAN™ protocol at the Requester can be extremely simple. Acquiring storage. Resolving the IP Address. Transferring block I/O. Releasing Storage. In the more complex extreme, it can acquire multiple partitions, be multi-threaded to each of those partitions, handle a semaphore relationship with other Requesters that share the same space, pass the token to other Requesters or establish disk Spanning and Mirroring.

- Name Resolver Level

  This is the protocol executed to resolve the Name of the Partition to the appropriate IP address. The IP address should be treated as a fluid variable due to reassignment through DHCP whereas the Name is a static element. The Resolver is called by the μSAN™ Protocol when it does not know the IP address of the partition or receives an error indicating the IP address may no longer be valid. The Name resolution protocol is proprietary to the μSAN™ and is discussed in "μSAN™ Name Resolution Protocol".

Block Transfer Protocol

The μSAN™ Block Transfer Protocol is a master/slave architecture with the Requester as the master. This protocol communicates with either the TCP or UDP transport layers. (It is assumed that the session has been established prior to executing this protocol.) There are nine basic commands. The format is set to operate under either UDP or TCP transport protocol and therefore defaults to the most common denominator. The first byte is the CMD byte and defines the subsequent data. The block size is fixed at 512 (or 530) bytes and the LBA (Logical Block Address) field is fixed at 40 bits. This limits the partition size to 512,000 GBytes. The protocol follows the "Big Endian" orientation (IP standard) for all but the Data Field. (The data field is Requester dependent and can be anything the Requester wishes.)

- ***Transfer Command***

  This command is used to transfer the data either as a write to the μSAN™ or the result of a request from the μSAN™. One block of data is transferred. In a write from Requester operation, this is the only command that is transferred from the Requester. The μSAN™ responds with an ACK Command. This command may be sent to either unicast or multicast destination IP addresses.

	Command	*Token*	*Delimiter*	*LBA*	*Data*
Size	1 Byte	X Bytes	1 Byte	5 Bytes	512* Bytes
Data (hex)	01	ASCII	0	Binary	Binary

** - 530 bytes if the block size has been extended*

- ***Request Command***

  This command is a request for a transfer from the μSAN™. The μSAN™ responds with a Transfer Command containing the requested data. No ACK Command is required in response to this transfer. This command may be sent to either unicast or multicast destination IP addresses.

	Command	*Token*	*Delimiter*	*LBA*
Size	1 Byte	X Bytes	1 Byte	5 Bytes
Data (hex)	02	ASCII	0	Binary

- ***Request Lock Command***

  This command is a request for a transfer from the µSAN™, however, it locks out all other accesses from any other authenticated Requester. The lock is released when a Transfer Command is received from this Requester or a time-out occurs. The timer for the time-out is retriggerable upon Request Command accesses from the initiating Requester. Time-out TBD. The function of this command is to provide for semaphore functionality of the data. This command may be sent to either unicast or multicast destination IP addresses.

	Command	*Token*	*Delimiter*	*LBA*
Size	1 Byte	X Bytes	1 Byte	5 Bytes
Data	03	ASCII	0	Binary

- ***ACK Command***

  This command acknowledges a successful transfer. It need only be used in when the communication is being executed through the UDP transport layer to recognize a successful Transfer command from a Requester to the µSAN™. Even in a TCP environment, the Requester needs to await an ACK before proceeding to insure single threaded error exception handling. The LBA that was received in the Transfer Command is sent to resolve the "Sorcerer's Apprentice" problem inherent in IP due to time-out retry. The µSAN™ IP address represents the address of the µSAN™ that responded to the transfer. (For example, a multicast GO TRANSFER command to a set of mirrored partitions would generate two ACKs from each µSAN™ back to the transferring µSAN™ who would then pass those ACKs back to the client for exception resolution. The ACKs that are passed back to the Requester will have the destination µSAN™s IP address.)

	Command	*LBA*	*µSAN™ IP*
Size	1 Byte	5 Bytes	4 Bytes
Data	04	Binary	Binary

- ***Error Command***

The Error Command indicates that an operation could not be completed for some reason and is a transfer from the μSAN™ to the Requester. The μSAN™ IP address represents the address of the μSAN™ that responded to the transfer. (For example, a multicast GO TRANSFER command to a set of mirrored partitions may generate an ERROR from one of the μSAN™ back to the transferring μSAN™ who would then pass that ERROR back to the client for exception resolution. The ERROR that is passed back to the Requester will have the destination μSAN™s IP address.)

Error Codes:

01	Invalid Authorization
02	Partition has locked you out
04	Go Command had an Invalid Authorization at the destination
08	Go Command was locked out of Partition at the destination
10	LBA out of Range
20	LBA is Write Protected

	Command	*LBA*	*μSAN™ IP*	*Error Code*	*Error Message*	*Delimiter*
Size	1 Byte	5 Bytes	4 Bytes	2 Bytes	X Bytes	1 Byte
Data	08	Binary	Binary	Binary	ASCII	0

Go Command Set

The "Go" commands are specifically used to copy a logical block from one μSAN™ partition to another μSAN™ partition. The Go commands are initiated by the Requester who resolves both of the partition's Names to their relative IP address and passes the command to one of the μSAN™s. The result is to double the effective bandwidth and to remove a burden from the Requester. This does not turn the μSAN™ into a master, the Name to IP resolution is still performed by the Requester and an ACK is communicated back to the Requester upon completion. The "Go" commands are not necessary for basic μSAN™ communication and should be considered optional on the μSAN™. Under these options are a few choices as to operation of the commands dealing with the transport layer utilized. The options would be identified in the μSAN™ personality. The following table shows those choices:

- TCP only — This default choice allows the μSAN™ to perform all Go commands as a TCP transport. Although TCP has more burden upon it, it will work in all network topographies.

- UDP only — A low-cost μSAN™ could be made that ONLY supports the UDP stack in the protocol, yet still supports the Go command

set. Use of UDP only is very effective in the LAN (or home) environment and perhaps is quite efficient in a single hop routed environment, however, it could quickly degrade in a multi-hoped internet environment.

- TCP/UDP selective — It is possible for the μSAN™ to determine an appropriate transport layer dynamically by comparing the destination IP with it's subnet mask. If the destination IP must be routed, then TCP is used. Otherwise UDP transport is used.

It should be noted that the Go commands may address another partition within the same μSAN™. In this special case, it is conceivable that the μSAN™ will execute the command internally without going to the physical layer for a substantial increase in performance. Indeed, the μSAN™ may execute the command as high as the μSAN™ application layer.

The Go command set may be used to transfer one partition to another virtual partition through multicast. (see "Use with Multicast IP") This perfectly acceptable. The ACK response is passed through to the Requester by the primary μSAN™'s ACK. If multiple ACK's are generated due to multiple μSAN™s responding (as in mirroring), multiple ACKs are generated for the Requester for command integrity.

- ***Go Transfer Command***

  The Go Transfer Command informs a μSAN™ to go active on the network and autonomously transfer a block of data to another μSAN™.

	Command	*Token1*	*Delimiter*	*LBA1*
Size	1 Byte	X Bytes	1 Byte	5 Bytes
Data	11	ASCII	0	Binary

(continued)

	IP2	*Token2*	*Delimiter*	*LBA2*	*Data*
Size	4 Byte	X Bytes	1 Byte	5 Bytes	512* Bytes
Data	Binary	ASCII	0	Binary	Binary

** - 530 bytes if the block size has been extended*

- ***Go Request Command***

  The Go Request Command is similar to the Go Transfer Command. The Requester issues this command to a µSAN™ to have it request data from another µSAN™. At the completion of the transfer, the µSAN™ will ACK or ERROR the Requester. TCP transport is used to send the request to the second µSAN™.

	Command	*Token1*	*Delimiter*	*LBA1*
Size	1 Byte	X Bytes	1 Byte	5 Bytes
Data	12	ASCII	0	Binary

(continued)

	IP2	*Token2*	*Delimiter*	*LBA2*
Size	4 Byte	X Bytes	1 Byte	5 Bytes
Data	Binary	ASCII	0	Binary

- ***Go Request Lock Command***

  The Go Request Lock Command is similar to the Go Request Command with the lock element added to it.

	Command	*Token1*	*Delimiter*	*LBA1*
Size	1 Byte	X Bytes	1 Byte	5 Bytes
Data	13	ASCII	0	Binary

(continued)

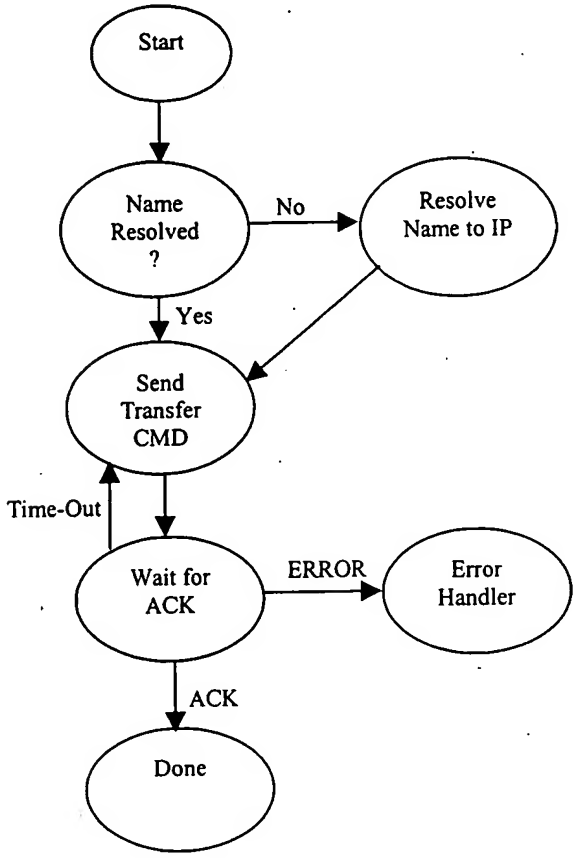
	IP2	*Token2*	*Delimiter*	*LBA2*
Size	4 Byte	X Bytes	1 Byte	5 Bytes
Data	Binary	ASCII	0	Binary

- ***Release Partition Command***

  This command will cause the µSAN™ to release the partition, erase the data blocks, and release the IP/Name placing the blocks back into the root pool for future allocation.

	Command	*Token*	*Delimiter*
Size	1 Byte	X Bytes	1 Byte
Data	0F	ASCII	0

State Diagram for a Requester to µSAN™ Transfer (Requester's Point of View)



μSAN™ Name Resolution Protocol

It is assumed (though not always the case) that the IP address of each device on the network is temporary. This is certainly true if the IP address was established through a DHCP server. Name resolution deals with the task of resolving a unique name in the network to a specific IP address. There are currently a number of Name resolution protocols in practice. Some resolve only in a LAN while others resolve throughout an internetwork and one resolves the Internet. All of these name resolution protocols place an unacceptable burden on the μSAN™.

- DNS (Domain Name Services)

  This Name Resolution protocol would be ideal, however, it requires the presence of a DNS server on the LAN to operate. This is typically not present in home or peer-to-peer network. Those networks that communicate to the Internet do so through a DNS server, however, this server is not accessible by a private network.

- WINS (Windows Name Services)

  This proprietary protocol implemented by Microsoft also require that a WINS server be present on the LAN that is not common in the home.

- NetBIOS over IP

  NetBIOS over IP Name Resolution is a peer-to-peer naming resolution protocol, however, it goes too far and places the resolved name into the Master Browser lists and requires that the NetBIOS protocol be implemented at both the Requester and μSAN™ level.

- LM HOSTS and HOST files

  These implementations are static and therefore not acceptable.

- SNS (Storage Name Services)

  This protocol is under consideration by the engineering groups as a new standard for IP storage. However, this protocol is being designed to resolve the issues of iSCSI and iFCP in an enterprise environment. The protocol is not peer-to-peer based and still requires a server for resolution. This certainly NOT a home network solution.

Clearly the μSAN™ needs it's own Name Resolution protocol. In the IP protocol, Name Resolution is under the domain of the application level and therefore may be part of the μSAN™ protocol. The requirements are that it be simple to implement and peer-to-peer. It need not extend itself through a router. One must be able to use proxy services to

translate the μSAN™ Name Resolution protocol to and internetwork protocol such as DNS or WINS.

The μSNR (μSAN™ Name Resolution) protocol uses a broadcast IP and determinate UDP port to resolve the Name very similar to NetBIOS over IP. As in NetBIOS, the Name is limited to 16 bytes. Once the Name is resolved, it is maintained in a cache at the Requester. The cache is used until it becomes invalid through an error response. Names must be unique on the network and it is up to the Requester to insure that no two same names are used accidentally. This may be performed by first interrogating the network for a name prior to partition allocation. **(Note: There is a potential window of error here that needs to be resolved.)**

The Requester establishes the Name during the process of partition allocation. The μSAN™ acquires an IP address from the DHCP server and assigns it to that name, but the Requester doesn't know what IP address that is. The name must be resolved for each transfer or request from the Requester to the μSAN™. This is performed in the following fashion (Requester point of view):

- Check to see if the name has already been resolved. If yes, use that IP. If no, place a call to the μSAN™ Resolver.

  Resolver

  1. Issue a UDP packet with a broadcast IP address to port XX with the Name to be resolved.
  2. All μSAN™ examine this broadcast message and see if the Name is on their drive. All other devices on the network throw away the packet.
  3. The μSAN™ that has the Name responds to the Requester at the requested port with the IP address. The Requester places this address in its cache.
  4. If there is no response to the request the Requester must resolve the error. (The μSAN™ may be turned off.)

μSAN™ Instantiation on the Network

The μSAN™ (or many μSAN™) need to be found at the root level when a Requester wants memory. This is performed by the μSAN™ FIND broadcast packet. This packet is issued as a UDP packet to the IP broadcast address to port XX. All μSAN™ respond to this request with a μSAN™ FIND RESPONSE pack to the requesting IP/PORT. The Requester must wait an appropriate period of time to receive all responses. Note that the root has no Name … it doesn't need one. This re-enforces the peer-to-peer nature of the μSAN™.

Broadcast Protocol

The µSAN™ uses a broadcast IP protocol with the UDP transport protocol to perform the functions of Name Resolution and µSAN™ Find. The following identifies these packets. It should be noted that the request/response sequences should be issued multiple times to insure no packets are lost/dropped.

- µSAN™ Find

  This request is issued by the Requester looking for the root IP of all µSAN™ on the LAN. Every µSAN™ that receives this packet must respond with µSAN™ Find Response packet to the calling IP/UDP Port. The µSAN™ Find packet is addressed to UDP Port XX.

	Command
Size	1 Byte
Data	80

- µSAN™ Find Response

  Each µSAN™ in response to a µSAN™ Find packet responds to the Requester with its IP address in a µSAN™ Find Response packet. This packet is addressed to the requesting IP/UDP Port. It is understood that there may be many responses to the request issued on the network and will be arbitrated at the network layer.

	Command	*IP Addr*
Size	1 Byte	4 Bytes
Data	81	Binary

- Name Resolution Request

  The µSAN™ Requester issues this packet to UDP Port XX as a request for Name to IP resolution. The µSAN™ that recognizes this Name as one of its partitions responds with a Name Resolution Response Packet.

	Command	*Name*
Size	1 Byte	16 Bytes
Data	90	ASCII

- Name Resolution Response

  The µSAN™ issues this packet in response to the Name Resolution Request packet with the IP address associated with the Name. It is issued to the requesting IP/UDP Port. The name is resent in case of multiple requests and the "Sorcerer's Apprentice Syndrome" present in UDP.

	Command	*Name*	*IP Addr*
Size	1 Byte	16 Bytes	4 Byte
Data	91	ASCII	Binary

µSAN™ Root IP Status and Control

It is at the root IP that one can get general information about the drive and to allocate a partition. These operations are performed by executing Block transfer commands to/from the root IP as shown. Note the Token field is a No-Op for these operations and there is no authentication for the Root IP.

- Allocate a Partition (Block Transfer Command)

  LBA = 0
  Token = ~~
  Data bytes 0 – 511
  a. Name
  b. Token
  c. ID Character String
  d. Authentication Tags
  e. Partition Size
  f. Personality Tags

- Get µSAN™ Root Status (Request Transfer Command)

  LBA = 0
  Token = ~~
  Data bytes 0 – 511

  a. µSAN™ Version
  b. µSAN™ Total Capacity
  c. µSAN™ Available Capacity
  d. µSAN™ Speed
  e. µSAN™ Reliability
  f. µSAN™ Portability
  g. µSAN™ QoS Capability

- Get µSAN™ Allocation Status (Request Transfer Command)

  LBA = 1
  Token = ~~
  Data bytes 0 – 511

  For Each Allocated Partition

  a. ID Character String
  b. Size

μSAN™ Performance Analysis

Only the block transfers are important in the performance of the μSAN™. The broadcast packets are seldom used and have no impact. This analysis assumes a 100Mb Ethernet network layer and the UDP transport layer.

- Transfer CMD Overhead

  The μSAN™ Transfer CMD header is made up of 23 bytes (assuming a 16 byte Token)
  The UDP header is made up of 8 bytes.
  The IP header is made up of 20 bytes

  Subtotal = 51 bytes (>46, No Ethernet Padding)

  The Ethernet head & tail is made up of 26 bytes.

  Total overhead = 77 bytes.

  With a 512 byte transfer, this implies an efficiency of (512/589) = 87%

- ACK CMD Overhead

  The μSAN™ ACK CMD is 10 bytes
  The UDP header is made up of 8 bytes.
  The IP header is made up of 20 bytes.

  Subtotal = 38 Bytes (<46 bytes, Ethernet Pad is added)

  The Ethernet head & tail is made up of 26 bytes.
  Need to add an Ethernet pad of 13 bytes

  Total Number of bytes = 72

- Request CMD Overhead

  The μSAN™ Request CMD is 23 bytes (assuming a 16 byte Token)
  The UDP header is made up of 8 bytes.
  The IP header is made up of 20 bytes.

  Subtotal = 51 bytes (>46, No Ethernet Pad needed)

  The Ethernet head & tail is made up of 26 bytes.

Total Number of bytes = 77

- Ethernet Inter-frame Gap

  The inter-frame gap for Ethernet (Between transmissions) is specified at 12 bytes.

- The maximum write to the drive (assuming no drive latency, no collision) of 1MByte from a Requester is:

  Number of Transmit/Ack packet pairs = 2000

  Transmit/Ack Handshake = (Transmit) + Gap + (Ack) + Gap
  = (589) + 12 + (72) + 12
  = 685 Bytes
  = 685 * (10/100) * 10^{-6} (Sec) (Assuming 100Bt)
  = 68.5µs

  2000 * 68.5µs = 137.0ms

  This implies a sustained maximum throughput of 7.30 MBytes/Sec

  (Note that with a 1GigaByte Ethernet Network, the maximum throughput would be 73.10 MBytes/Sec)

- The maximum read from drive (assuming no latency, no collision) of 1MByte from the µSAN™ to the Requester is:

  Number of Request/Transmit packet pairs = 2000

  Request/Transmit Handshake = (Request) + Gap + (Transmit) + Gap
  = (77) + 12 + (589) + 12
  = 690 Bytes
  = 690 * (10/100) * 10^{-6} (Sec) Assuming 100Bt)
  = 69.0µs

  2000 * 69.0µs = 138.0ms

  This implies a sustained throughput of 7.25 MBytes/Sec

  (Note that with a 1GigaByte Ethernet Network, the maximum throughput would be 72.5 MBytes/Sec)

Multi-user and Multi-session Resolution

The μSAN™ must answer to many devices. Each partition must be treated mutually exclusive to the other partitions (there may be an exception to this when dealing with QoS). Each Requester needs to choose whether it will communicate to the μSAN™ in either a UDP or TCP transport protocol. The advantage to UDP is simplicity and speed in a LAN environment. TCP may become the transport of choice if the Requester is communicating through a router and/or is multi-sessioned with the need for the μSAN™ to have many sessions open simultaneously. The μSAN™ supports both UDP and TCP transports utilizing the same application level protocol. The advantage of "sliding windows" in the TCP protocol is not used due to the lack of a many block transfer command (which would complicate exception handling).

A session is identified by the μSAN™ by the ID/Port pairs used in the communication process. TCP has the additional advantage of instantiating a session prior to the data transmission whether the port is reassigned on the μSAN™ to an "ephemeral" port. Allowing another session to be started by the same Requester with out terminating the previous session.

Consider UDP

> The Requester application selects a UDP port (this can be any port it chooses, but by convention should be one of the ephemeral ports (1024-5000). It then executes a command to the μSAN™ at the partition IP address and the μSAN™ FIXED UDP port address (not yet established but will most likely be one >5000). The μSAN™ will consider this to be a session with the identifying features of Requester IP/(ephemeral PORT) and μSAN™ IP/(fixed PORT). If a Requester wishes to execute a second session without completing the first, it must use a different ephemeral PORT to avoid contention with the first session.

Consider TCP

> The Requester application selects a TCP port (again this is may be any port it chooses). It then establishes a TCP session with the μSAN™ by first communicating to it through the FIX TCP PORT address (not yet established but most likely to be one >5000). This is part of the TCP layer and is outside of the μSAN™ protocol. During this instantiation, the μSAN™ TCP Layer will re-assign the μSAN™ PORT address to an ephemeral port address. This session will then be identified by the Requester IP/(ephemeral PORT) and the μSAN™ IP/(ephemeral PORT) pairs.

Semaphore/Lock Support

> The μSAN™ has a set of LOCK commands to be used in multi-session and/or multi-user environments. Theses commands are initiated by a Requester read

activity in anticipation of a following write. The commands allow a read to be followed by a write as an atomic operation (without intervention by another session or user). The IP/PORT pairs (as discussed above) are used to identify who can access the partition following a REQUEST LOCK COMAND. The partition is unlocked upon a valid write to the partition.

Quality of Service Resolution

QoS is dependent not only on the parameters of the μSAN™ but the underlying network and its topology. If the underlying network supports some form of QoS, the μSAN™ will take full advantage of it. For example, if Ethernet is the network in use, then the μSAN™ will set the Type of Service bits in the Ethernet Header to maximum transfer.

There is an opportunity for the μSAN™ to arbitrated and guarantee its bandwidth.

Impact Upon Physical Network Layer

The μSAN™ places some special needs upon the network topology. It has one MAC address associated with many IP addresses. The DHCP server should be programmed to allow for the growth in allocated IP. (This is not a problem behind a NAT router.)

It should be recognized that in the network topology, the μSAN™ should be allocated the most bandwidth. Ideally, it should be connected to a switch, not a hub. Multiple μSAN™s on the same network will then be allocated full simultaneous bandwidth provided that they are not communicating with the same Requester.

The μSAN™ packets are small enough that the packet will not be fragmented when communicating through Ethernet (MTU = 1492 bytes), however, when communicating in point-to-point protocols fragmentation may occur. (This will only impact performance. Data integrity will be maintained.)

Impact upon the IP Layer

At the Requester … Standard IP

At the μSAN™ it is considered to be multi-homed from an IP point of view, but since it has only one interface (one MAC) it receives only one packet at a time. The μSAN™ must be programmed as re-entrant code dependent upon the IP/Port pairs (Requester and μSAN™).

The concept of one MAC and many IP places some additional burden upon the μSAN™ when considering ARP, DHCP, ICMP and IGMP.

- The μSAN™ ARP (Address Resolution Protocol) code must be capable of associating many IP addresses with one MAC address. (It may share the same ARP cache.)
- The μSAN™ must be capable of sustaining multiple DHCP allocations and timers (sec resolutions), one for each IP address. The μSAN™ must use the unique identifier option for association because there is only one MAC.
- ICMP must respond to multiple IP addresses
- IGMP must maintain multiple multicast IP, each allocated to a different partition.

Impact upon UDP Layer

At the Requester

- Use fixed μSAN™ Port address for all communication the μSAN™s.
- Standard Protocol

At the μSAN™

- Multi-threaded dependent upon IP/Port quads.
- Needs an assigned Port address (>5000). This address will be used for Name Resolution, μSAN™ Search, and μSAN™ Transfer.

Impact upon TCP Layer

At the Requester

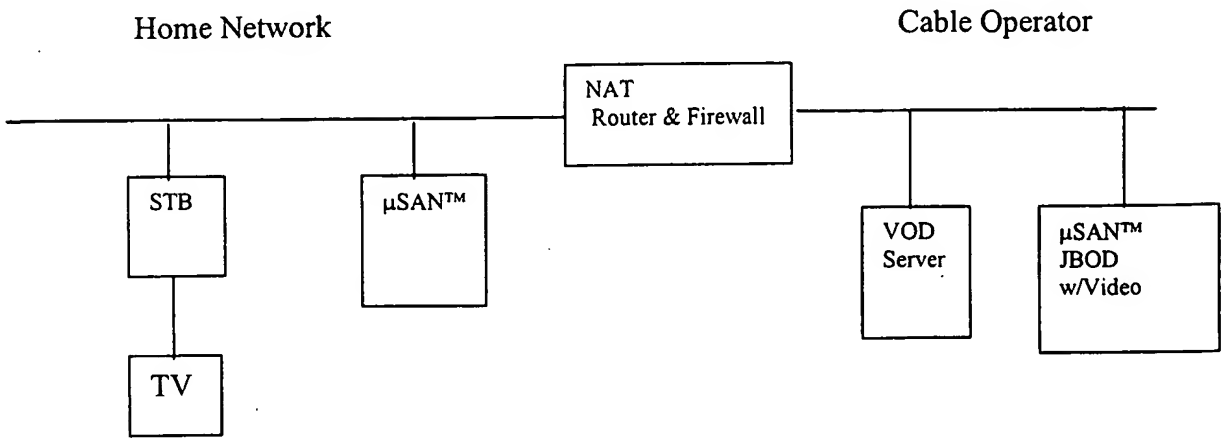
- Use fixed μSAN™ Port address for all session instantiation with the μSAN™s.
- Standard Protocol

At the μSAN™

- Multi-threaded dependent upon IP/Port quads.
- Needs an assigned Port address (>5000). This address will be only be used for μSAN™ Transfer.

Use in a NAT Firewall Environment

There are three commands that allow the µSAN™ to copy data from another µSAN™ at the direction of a Requester. These commands were specifically designed to transfer across a NAT router environment without opening a port in the firewall. Consider the following network topology.



Here we have a network compliant set top box that requests a movie from the MSO. The request is handled by the VOD Server who authorizes the transfer from the MSO's µSAN™ JBOD video repository. On the home network side, the STB is the Requester and informs his µSAN™ to GO REQUEST a block of data from the JBOD. This saves bandwidth by allowing the µSAN™ to get the data rather that having it moved twice (JBOD to STB, STB to µSAN™) in addition, because the µSAN™ initiates the transfer across the NAT, the NAT programs itself to allow the response from the JBOD to get back to the µSAN™ without having to program an open port.

Use in a WINS or DNS Environment

The μSAN™ Name Resolution protocol will only work in the LAN (due to broadcast IP protocol) and not traverse across the router. In those situations where the μSAN™ must be found in an internetwork (multi-networked) environment, the Name must be resolved by either a DNS or WINS server. This is possible through a proxy service residing on a PC or server located on the LAN. Using a standard NIC (Network Interface Card), and some application level software, this service may find μSAN™ resources and have them placed into the DNS or WINS tables.

Use in a Windows/PC Environment

There are at least two implementations of the μSAN™ in the Windows/PC environment.

- PC Communicating to a μSAN™ as a Requester (or proxy).

  Because the μSAN™ is a block level storage device, the PC may attach to it, as it would to any direct attached drive. The μSAN™ Name Resolution is out side of NetBios and so there is no conflict. All that is necessary is driver and the μSAN™ will appear as a drive in the Windows OS. The partition will not appear as a network resource to the windows environment until the user elects to "share" the partition or any of its components (folders or files).

  It is interesting to note that the locality of the μSAN™ may be in the PC itself and it is possible to boot from the μSAN™.

- PC Behaving as a μSAN™

  A PC may be programmed to behave as if it is a μSAN™. This would provide the same functionality to the network Requesters. Naturally, this low-cost implementation carries with it the burdens of power-on and OS dependency. The NIC and IP stack protocol would also need to be capable of one MAC with many IP addresses.

Use in a PC/MacIntosh, Linux/Unix Environment

The μSAN™ is NOT tied to the WIN/OS. This allows for the μSAN™ to be used with WIN/PCs, MAC/PCs, Linux/Unix Workstations, etc.

Use by a Low-Cost Appliance Device

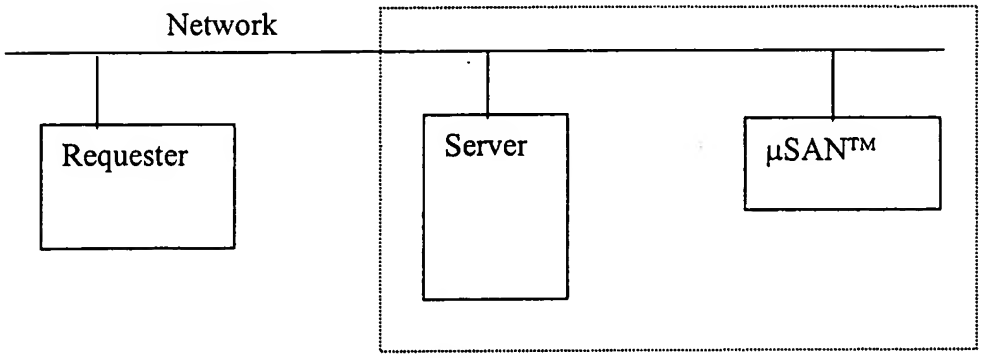
The μSAN™ was designed to support low-cost Requester implementations. At a minimum the Requester only needs to support the IP stack above the UDP transport level. This represents a very simplistic code architecture and implementation. The application layer code sitting above UDP will perform such operations as:

- Discovery
- Allocation/De-allocation
- Name Resolution
- Block Transfer
- Exception Handling

Sitting above the μSAN™ application layer would be appliance specific needs such as partition format/file system, sharing through token passing and data management.

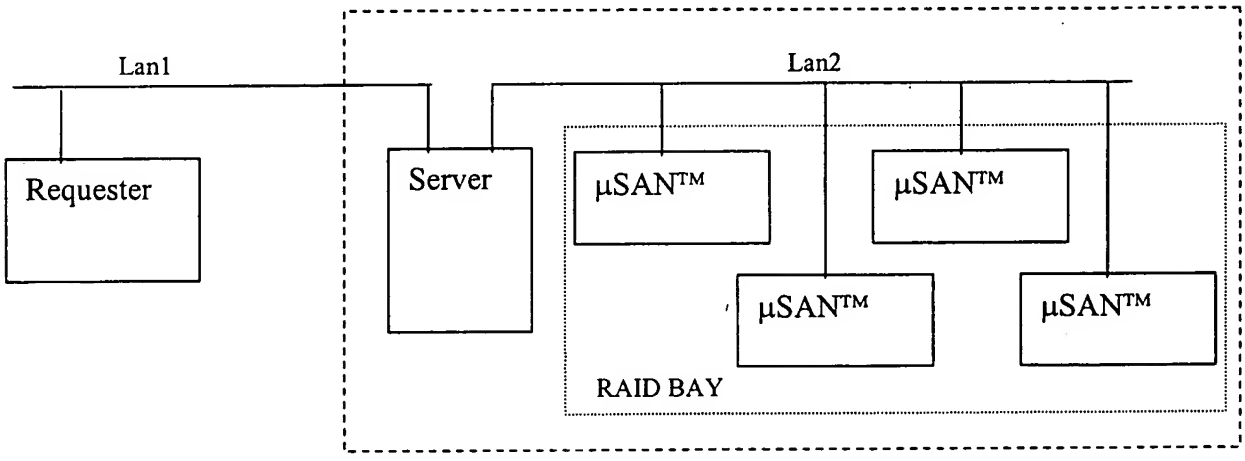
Use in a Server Environment

The µSAN™ will work well in a client/server environment in addition to a peer-to-peer topology. Consider the following:



In this network, the Server has grabbed all of the µSAN™, partitioning it and establishing authority. The Requester finds a µSAN™ and the Server responds saying that it can partition the space. The Server takes the name, and associates it with one of the established partitions. The Requester then communicates directly to the µSAN™. Note that the server retains authority over the partition making this a client/server environment.

The server could be multi-homed (multi-networked) with the µSAN™ established on a private LAN as follows. In this situation, the server proxies all of the µSAN™ commands from Requester allocating the µSAN™ network at its discretion. It this situation, the server may behave as a cache and also provide RAID services.



Use with Multi-Cast IP (Spanning & Mirroring)

Background

Multicasting in IP was created to serve the need to send single packets to multiple recipients in an internetworking environment. Unicast is a point-to-point communication and therefore dictates that there be a separate packet for each recipient. Broadcast IP is limited in scope to the LAN and burdens that LAN's hosts with unnecessary filter processing. Only multicast IP fulfills this need. However, multicast IP has some complications not associated with either unicast or broadcast and a protocol was developed at the IP stack level known as IGMP (Internet Group Management Protocol) to handle these issues. IGMP performs such functions as Group Creation (including IP address assignment), Group Joining and Group Release. This paper will not go into the implementation of this protocol, however, it should be understood that both μSAN™ clients as well as μSAN™ storage elements need to support this protocol for IP multicast compatibility as well as the support of multicast compatible routers. (Note that this protocol is quite light for both source and recipient. IGMP, however, is quite burdensome for routers.)

Multicast does not share the same disassociation that unicast IP and MAC have. There is a direct correlation between multicast IP and multicast MAC. Indeed, in the Ethernet world the addresses are exactly the same for the lower 24 bits. Whereas a unicast MAC address is fixed, the multicast MAC is dependent upon the IP address and is transitory. This has significant impact on the implementation of multicast partitions in the μSAN™. Under unicast there was one MAC with many IP. Under multicast, there is a multicast MAC for each partition. Either the MAC filter at the physical stack layer has to be more flexible or all multicast (class D) packets are received and filtered at the IP layer.

Not all physical networks support multicast. In these situations, multicast packets are converted to broadcast packets at the bridge router and are filtered at the IP layer.

Each partition must support multicast independently. Therefore additional protocol is needed. Multicast is also a protocol that may be implemented (and changed) after partition instantiation. The additions to the μSAN™ protocol are therefore directed in unicast to the specific partition IP and are not a part of instantiation (other than partition personality).

Some Uses of Multicast Support

It is possible to use Multicast through the μSAN™ protocol to support RAID functionality. This paper shows how Multicast support for the μSAN™ cleanly enables two very powerful and desirable storage constructs; "mirroring" and "spanning". Disk mirroring is where one disk or partition exactly duplicates the storage of another with the advantage of dynamic backup and sometimes increased performance. Disk spanning

allows for two or more disk drives (or partitions) to appear to the computer as if it is one partition.

Let's look at mirroring. Suppose a network has two IP unicast IP partitions (preferably on separate drives. Bind the two partitions with a single multicast IP and each transfer to the drives goes through multicast IP. Each transfer from the drive is selected through unicast (to one of the drives). The result is mirroring.

Let's look at spanning. Again suppose a network has two IP unicast partitions bound with a single multicast IP. Further, suppose that the second drive has an offset LBA equal to the LBA size of the first drive. Each transfer to the drive is multicast and each drive looks at the LBA to determine if the transfer goes to that drive. The drive that doesn't want it, ignores it, while the other drive stores the data. On transfers from the drive, only the drive that has the LBA range answers the GET request.

Note that this protocol works through routers as well enabling disk farms for backup.

Partition Personality Variables

It is anticipated that not all µSAN™s will be multicast compliant. Further, each µSAN™ drive may be limited in how many of its partitions may support multicasting. Personality variables will need to reflect this.

Block Transfer Protocol

The additions to the protocol for multicast support are shown in the Block Transfer Protocol class simply because they are directed to the unicast partition specifically.

- ***Set multicast IP Command***

  This command will cause the µSAN™ to cause the partition to enable multicast capability to this partition at the specified multicast IP address. Based upon the physical network interface, the µSAN™ will translate this address appropriately to the multicast MAC. (Note: It may be possible for a µSAN™ to support more than one multicast address.) Upon receipt of this command, the µSAN™ will issue a IGMP JOIN message and will respond to IGMP QUIRES to this multicast address. The µSAN™ will respond to the

	Command	*Token*	*Delimiter*	*Multicast IP*
Size	1 Byte	X Bytes	1 Byte	4 Bytes
Data	09	ASCII	0	Binary

- ***Release multicast IP Command***

  This command will cause the μSAN™ to cause the partition to release the multicast IP associated to this partition. (Note: It may be possible for a μSAN™ to support more than one multicast address.) The μSAN™ will no longer answer multicast IP to this address nor will it respond to an IGMP QUERY.

	Command	*Token*	*Delimiter*	*Multicast IP*
Size	1 Byte	X Bytes	1 Byte	4 Bytes
Data	0A	ASCII	0	Binary

- ***Set LBA Offset Command***

  This command though used in multicast, is independent of multicast status. This command will set the LBA relative starting address for the partition (used in spanning).

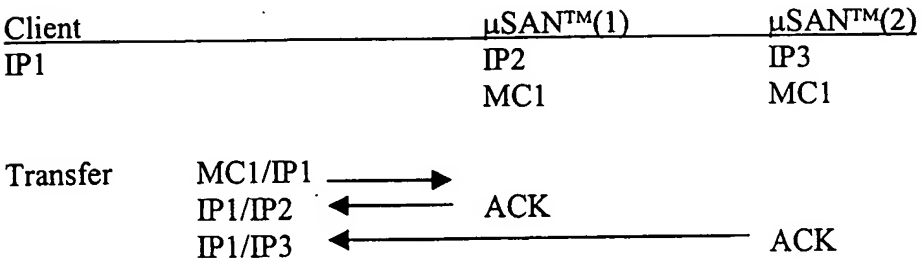
	Command	*Token*	*Delimiter*	*LBA Offset*
Size	1 Byte	X Bytes	1 Byte	4 Bytes
Data	0B	ASCII	0	Binary

Client to μSAN™ Transfer Protocol

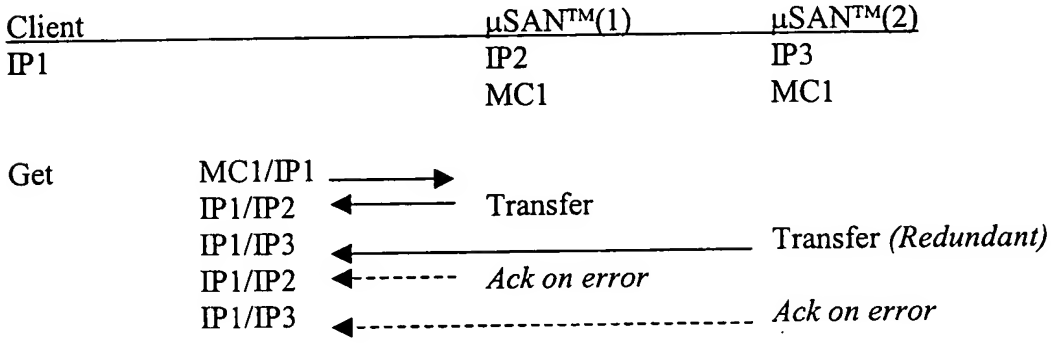
The constructs for client to μSAN™ need to be defined for appropriate operation. (IP refers to unicast addressing while MC refers to multicast addressing.) It should be noted that IP protocol prohibits the used of a multicast address in the source field of the IP packet header. This prohibits the μSAN™ as a target component from performing a μSAN™ to many client operation.

Consider Mirroring:

Upon each transfer to the µSAN™'s, each µSAN™ must answer with a unicast ACK. It is up to the client to wait for the appropriate ACKs and to handle any exceptions with unicast IP to that µSAN™.

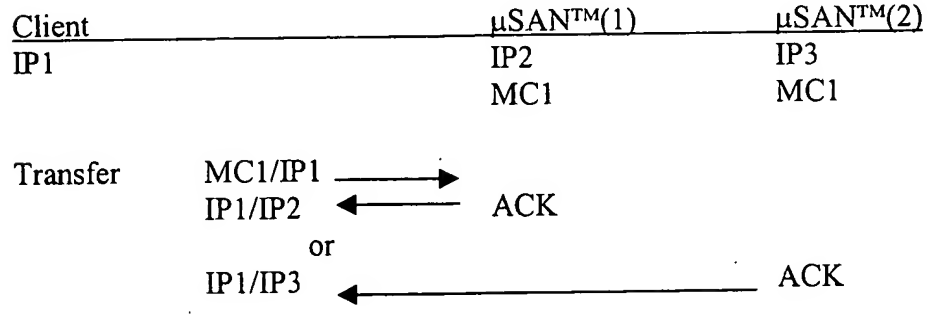


All transfers from the µSAN™'s would most likely be in the form of unicast, however, it is perceivable that a multicast GET could be performed and the client would simply "ignore" the redundant extra transfer. (For performance reasons on a high speed net.)

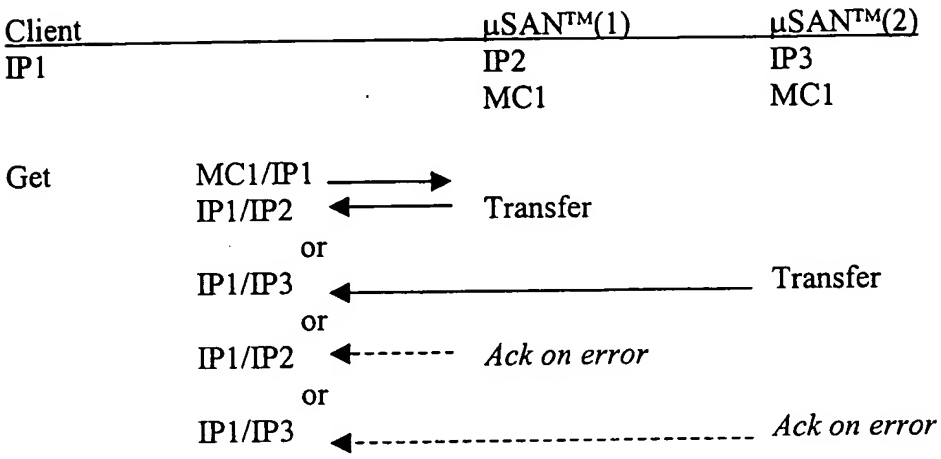


Consider Spanning:

Upon each transfer to the µSAN™'s, each µSAN™ must answer with a unicast ACK. However, because of the offset LBA, only one µSAN™ WILL answer.



A multicast GET would also be further selected by the LBA offset.

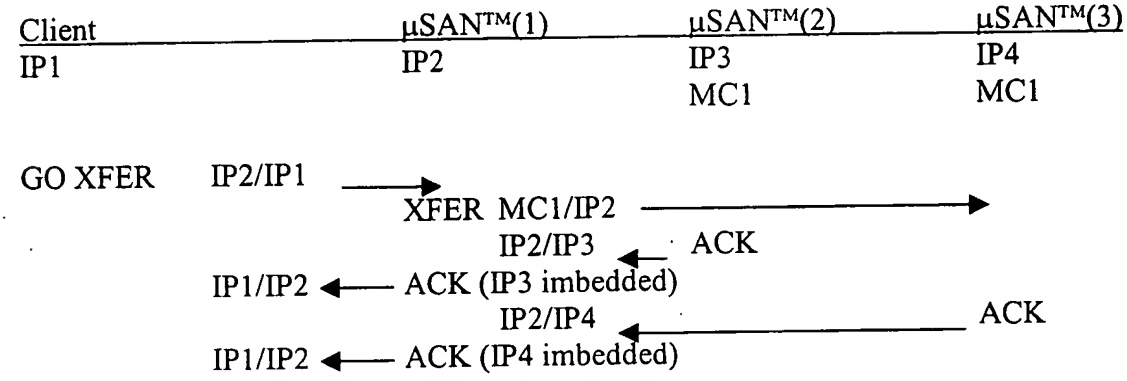


µSAN™ to µSAN™ Transfer Protocol

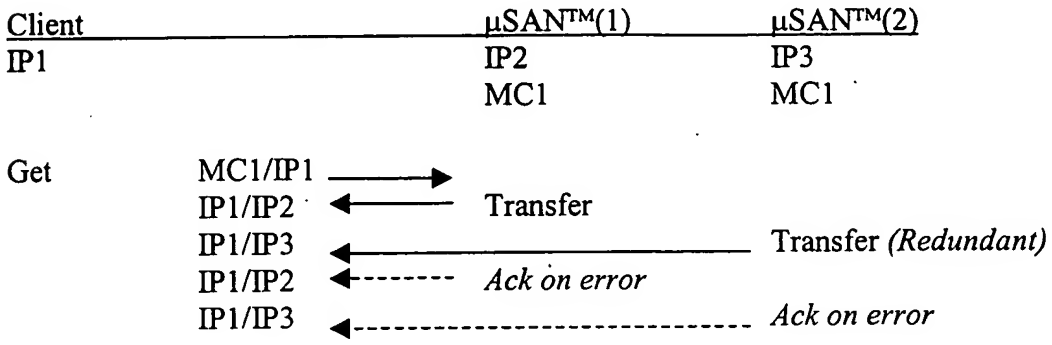
The constructs for µSAN™ to µSAN™ multicast transfers through the GO commands is a bit more complicated and needs to be defined for appropriate operation. It should be understood that the client is the only intelligence that knows when an exception occurs and how to handle it. Therefore it requires information in the ACK/ERROR response that shows which µSAN™ generated it. (Hence, the IP address in the ACK and ERROR command.)

Consider a µSAN™ copy to a Mirrored Set of µSAN™s:

It is assumed that the requester has previously established the multicast addresses and setup the mirrored partitions accordingly. It also knows the unicast IP of each of the mirrored partitions. The µSAN™ that sends the copy is unaware of the network topology, nor does it care about exception handling. These burdens are placed upon the Requester.

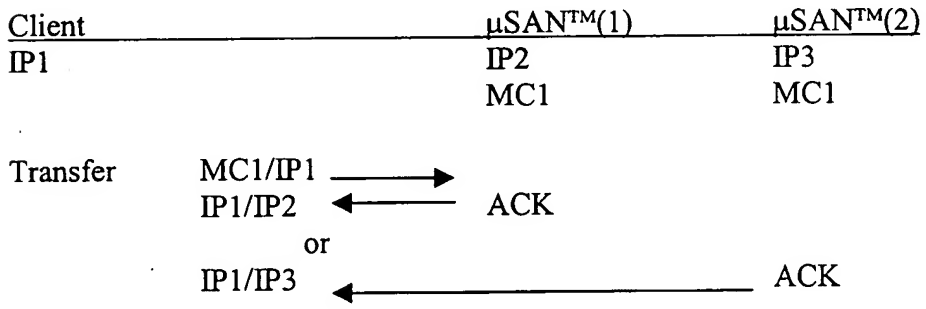


All transfers from the µSAN™'s would most likely be in the form of unicast, however, it is perceivable that a multicast GET could be performed and the client would simply "ignore" the redundant extra transfer. (For performance reasons on a high speed net.)

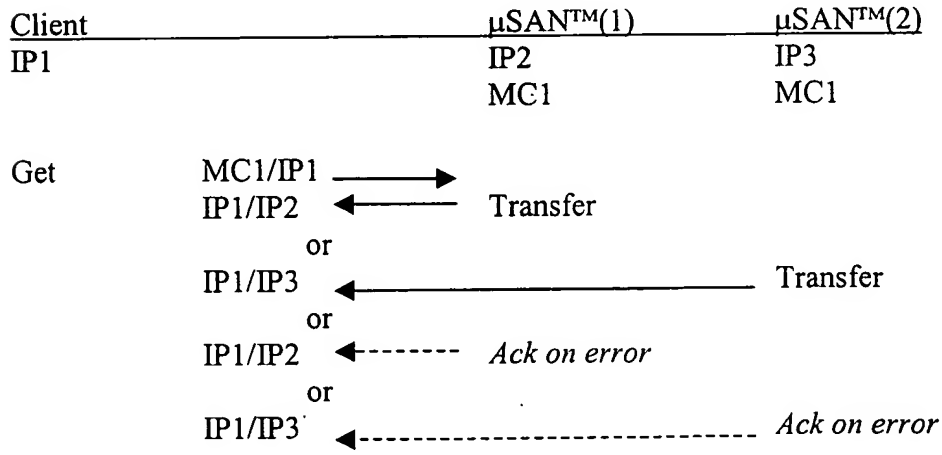


Consider Spanning:

Upon each transfer to the µSAN™'s, each µSAN™ must answer with a unicast ACK. However, because of the offset LBA, only one µSAN™ WILL answer.



A multicast GET would also be further selected by the LBA offset.



Considerations of Using the μSAN™ with UPnP

Universal Plug and Play is an open standard for the instantiation and communication of variables and events for appliances on an IP based network. This standard specifies the methodology of:

- Establishing IP
- How to deal (or not deal with DNS)
- How appliances announce themselves upon the network
- Notification of "events"
- How appliance "setup" variables are dealt with.

(Synonymous with Plug and Play on a computer system.)

Currently there are no Working Committees to consider network storage. Therefore the user variable, events and services are yet to be defined. This section identifies how the μSAN™ could be folded into the UPnP family of products. Note that a new Working Committee has been proposed to establish a generic set of variables with a class of vendor specific elements. It is quite possible that the μSAN™ may take advantage of this Basic DCP (Device Control Protocol).

To support UPnP, the μSAN™ must support at a minimum the following IP protocols:

- IP, UDP, TCP, IGMP, ICMP, DHCP (same as μSAN™ with the possible exception of TCP)
- HTTP
- HTTPU (on top of UDP)
- HTTPMU (on top of UDP with Multicast IP support)
- SOAP (Used for Remote Procedure Calls (RPC) to deliver control messages
- XML
- GENA (Generic Event Notification Architecture)
- SSDP (Simple Service Discovery Protocol)

UPnP provides for IP "Addressing", Appliance "Discovery", "Description", "Control", "Eventing" and "Presentation" (Optional). It is proposed that UPnP only address the root concerns of the μSAN™ and not the individual IP partitions. This both reduces the complexity of a large number of partitions as well as maintaining the mutual exclusivity of each partition to its Requester.

Those elements within the μSAN™ specification that may be handled through the UPnP protocol include:

- μSAN™ Discovery
- μSAN™ Root Status
- μSAN™ Partition Allocation

Addressing

This is the procedure used to obtain an IP address. UPnP specifies that the device first try to obtain an address from a DHCP server. If this fails (no server present), then it uses a procedure called "Auto-IP" where the appliance "intelligently chooses an IP address in the 169.254.X.X range". The device then periodically checks to see if a DHCP server becomes available to use those services instead. The impact upon the μSAN™ is minimal. Each μSAN™ partition and root will support the UPnP addressing scheme. The UPnP specification does not deal with Name conventions. Indeed it is benign to all Naming conventions and therefore will support the μSAN™ Naming convention for Name to IP resolution.

Discovery-Advertisement

The UPnP architecture states that a device must advertise its existence upon the network on a periodic basis. A time to live variable is included in the advertisement and a new advertisement must be transmitted before the previous advertisement dies or the control (Requester) may assume that the μSAN™ is no longer available. If a device is removed (power-down) from the network in a friendly way, then it must also send a message that it is going away. Multicast IP is used to send these messages upon a specific Multicast IP address (with a very short TTL header). The μSAN™ will only advertise its "root" IP and not each of the partitions.

Discovery-Search

The UPnP architecture states that a device must also listen for a discovery message from a Requester. This is also performed through Multicast IP. The Discovery Message includes search criteria that are used for comparison to determine if the μSAN™ will respond. The μSAN™ will only listen to criteria that deal with its "root".

Description

Discovery provides a URL to the Requester for a "Device Description" of the μSAN™ in XML. Included in this Device Description are URL's for a "Service Description", a "Presentation" and an enumeration of all services including their URL's for control and event. It is here that the basic DCP may be expanded to incorporate specific control and status of the μSAN™.

Presentation

A presentation page, (browser accessible), may be provided by the μSAN™ for specific control and status functions. It is perceived that these functions would relate only to the root.

Control

Control may be sent to the μSAN™ through SOAP to the Control URL identified during "description". Again, only the root may be controlled (possibly for allocation).

Eventing

It is possible to inform a μSAN™ to send out a notification if a specific event occurs. This is commanded by the Requester. Possible events at the root level are exception conditions such as overheating, bad power conditioning, sense an attack of "denial of service", etc. Events at the partition level are not supported.